

Project 2 - ArcSDE Enterprise Geodatabase Domain Validations

Project Description:

Design an FME utility that can dynamically identify all Esri ArcSDE Domain Violations across all database tables controlled by Esri Domain Rules

Project Date:

July 2021

Included Files:

Workspaces

File Name	Description
RecentEditDomainViolations.fmw	Domain Violation Identification Workspace

Excel Reports

File Name	Description
RH QC Dashboard Domain Violations.xlsx	Final output detailing feature classes and counts of invalid values across the entire database during the time range specified by the user. Names and Agencies have been stripped out.

Detailed Description:

As most people in the GIS Industry know, Esri is the *official GIS ecosystem* of government agencies, and in fact, of mostly everyone. Due to Esri's business model (only providing low-level tools and capabilities to its users while simultaneously locking more advanced features behind a hefty pay-wall) and the ubiquity with which the Esri software permeates nearly every GIS agency, a need was identified to develop a tool which would clean up the data fragments left behind in the Esri ArcSDE Geodatabase.

Domain violations occur when a value is entered into a field that violates the list of valid values as determined by the domain constraints. meant to enforce data consistency on a field-by-field basis. In this case, we are referring to Esri Geodatabase Domains and not a traditional SQL Domain. The Esri Domains defined in ArcCatalog can easily be violated by Esri's own toolset including the Field Calculator. Additionally, domain violations can also be input via programming/scripting and direct SQL access.

Source Data:

The entirety of the Arizona Department of Transportation's(ADOT) ArcSDE Enterprise Geodatabase(SQL Server). A heavy emphasis was placed on the 117 database tables representing Route Network and Route Events within ADOT's Esri Roads & Highways(RH) managed solution. The Esri RH solution is the authoritative LRS digital representation of Arizona's road network called the Arizona Transportation Information System(ATIS). This LRS Network is mission-critical and provides many data extracts required for submission to various federal programs including ARNOLD, FHWA, HMPS, etc. The Esri Geodatabase stores domain table entries in the form of XML which must be extracted before processing.

Proposed Solution:

Goal

Create an FME workspace capable of dynamically determining the domain violations across the database.

Constraints:

1. This workspace needs to be dynamic in the sense that as new domains are added, old domains are deleted, and database tables are added/removed, the process will be capable of continuing to run and validating the live data versus the current state of the domain tables.
2. Performance is a very large consideration. Each of the 100+ database tables can include hundreds of thousands or millions of records. Many tables also have upwards of 50 columns.

Process Overview

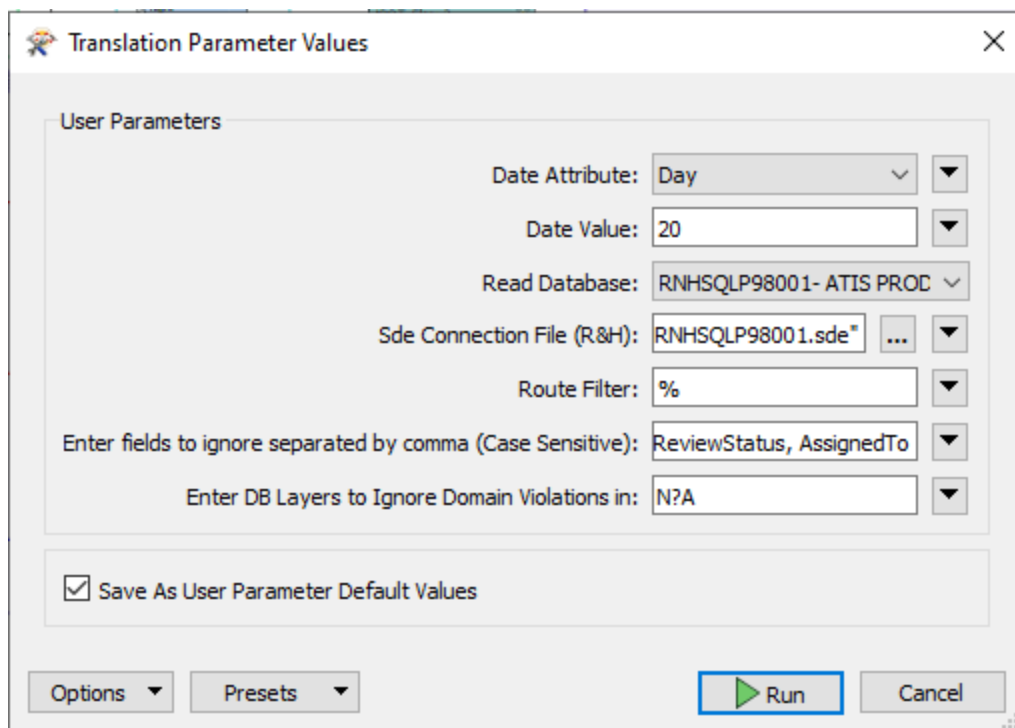
1. Using an FME SQL Executor, run a SQL query against the database to retrieve a list of all tables and their associated fields contained within the domain table
2. Use a series of FME transformers to create a single SQL statement that will read in records from every database table controlled by the domain in the XML definition.
3. Process the resulting data through a custom transformer
4. Publish the results to a Google Sheet which will serve as an authoritative Google Data Studio Dashboard. This dashboard will give analysts and managers insights into their database errors

FME Solution:

Input Dialog

The user input parameters, as seen below, allow the user to:

1. Specify a date dimension (year, month, week, day, hour) to look backwards into the database and query errors for a specific time range.
2. Specify the date value. In the screenshot below, the FME workspace would generate a report looking at the database domain violations entered within the last 20 days.
3. Select the SQL connection to the database, this is where we read the xml definition from and determine what to pass to the SDE reader
4. Select an SDE connection file, this connection only needs read privileges
5. This is a SQL query for route filtering, allowing us to target a specific route or set of routes. I will show how it's used in a custom transformer below
6. Optionally we can provide specific fields that we know have errors and we don't currently care about
7. Optionally we can specify entire feature classes to ignore in the database



Translation Parameter Values

User Parameters

Date Attribute: Day

Date Value: 20

Read Database: RNHSQLP98001- ATIS PROC

Sde Connection File (R&H): RNHSQLP98001.sde

Route Filter: %

Enter fields to ignore separated by comma (Case Sensitive): ReviewStatus, AssignedTo

Enter DB Layers to Ignore Domain Violations in: N?A

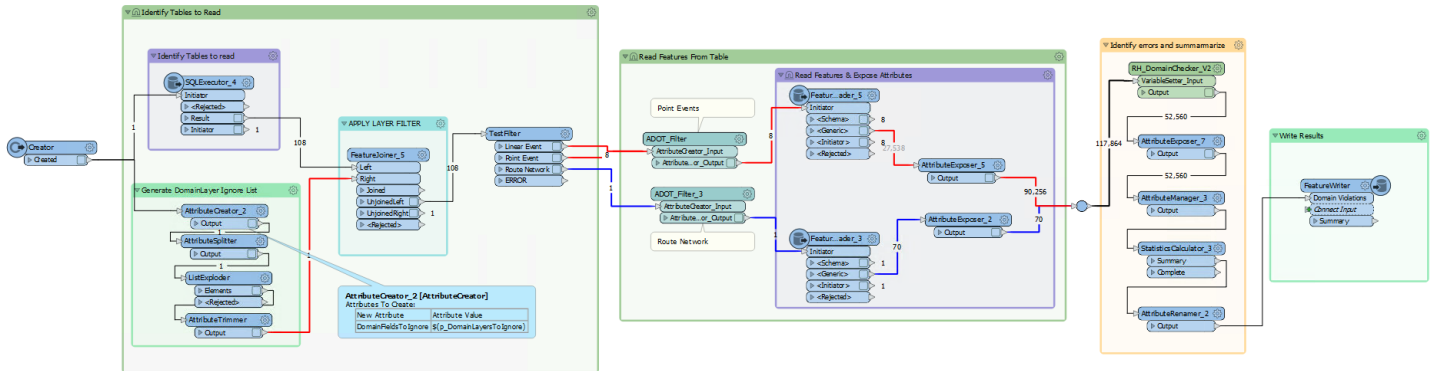
☒ Save As User Parameter Default Values

Options Presets Run Cancel

Main Overview

The entire process can be summarized in 4 steps:

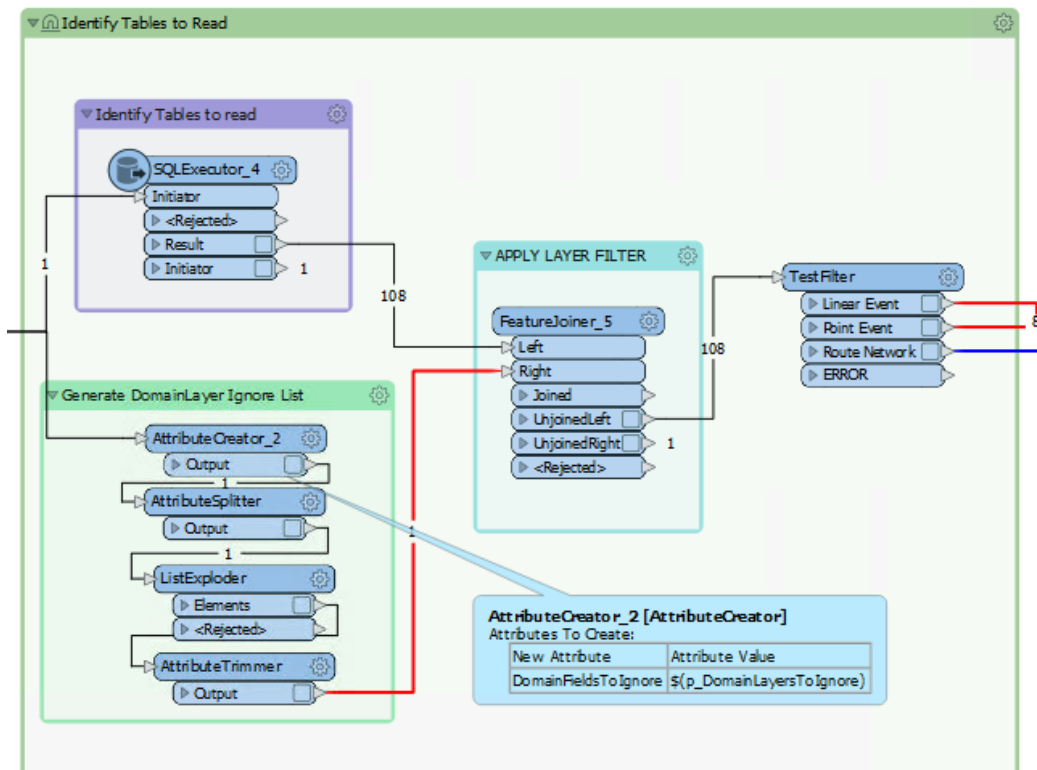
1. Identify the tables needing to be QCed
2. Read features from those tables
3. Identify errors
4. Summarize and write results



1) Identify Tables to Read

The process to identify which tables to read is comprised of the following steps:

1. Use a SQL query to identify all tables in the RH database as well as their type as registered in Esri Roads and Highways (Route Network, Linear Event, Point Event)
2. Create attributes of feature classes to ignore based on user parameter
3. Join all registered tables to table ignore list. Any joined records in this feature joiner are tables that the user has selected to ignore, only the unjoined left (tables registered in RH and not in the ignore list) will continue
4. Test Filter to separate features based on RH type. This is necessary as each type has specific requirements in the SQL query due to which fields exist in the RHG system.

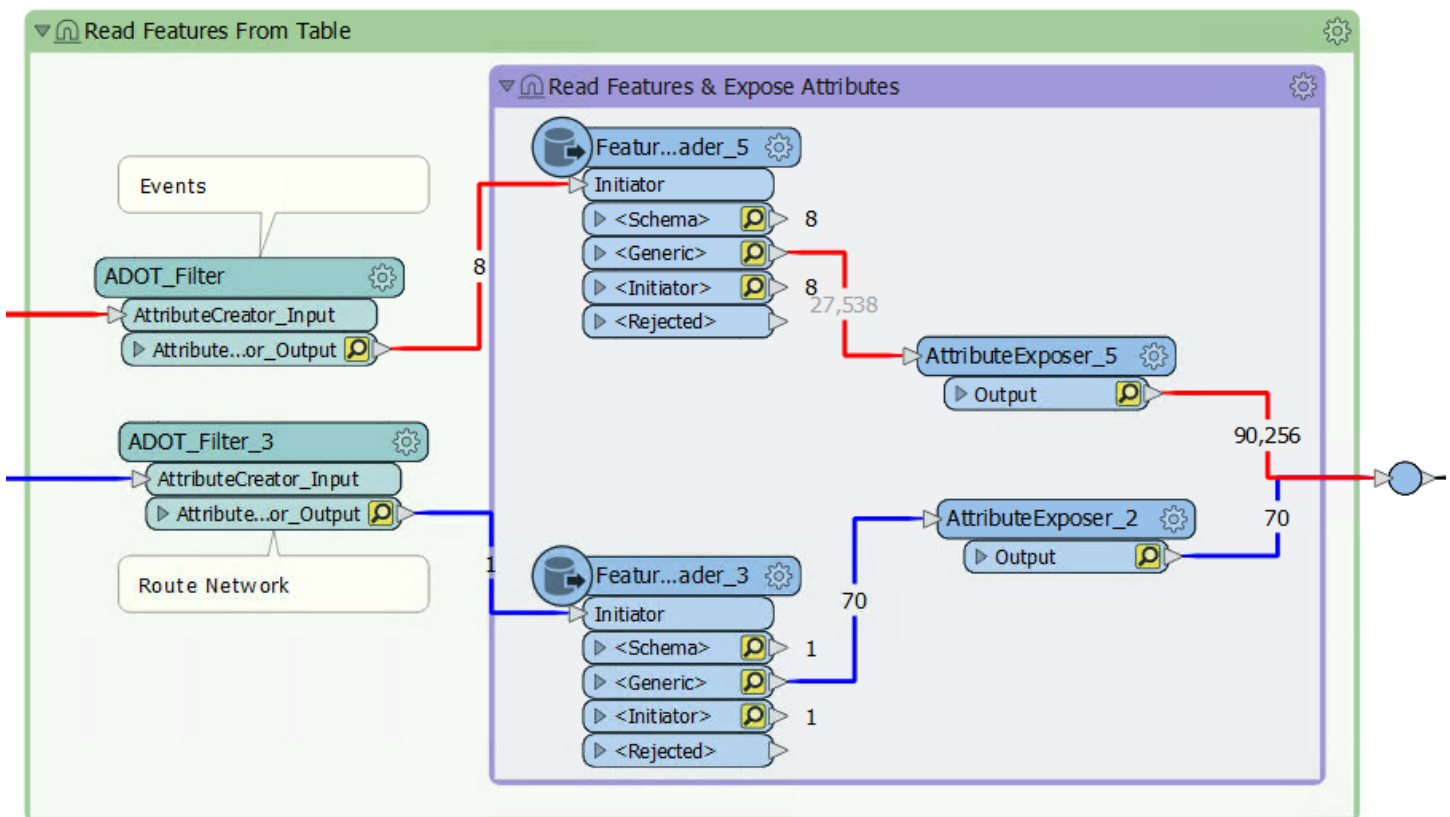


2) Read Features From Tables

1. The features that were previously separated into streams get passed into a custom transformer used in all my space. This custom transformer, ADOT_Filter, allows me to standardize SQL queries in one workspace and have them applied to all my processes simultaneously when I need to make a change. I'll talk about this more in the next section below.
2. Next each feature stream is passed into an Esri SDE Feature reader, reading features from the table based on
 - a. The time frame the user selected in parameters
 - b. The sql query generated by the ADOT filter, designed to standardize the SQL queries and results
 - c. The SQL query for the feature reader in Events looks like this:

```
1 @Value(Filter_Current) @Value(Filter_Event_Extra)
2 and (
3     last_edited_user IS NOT NULL
4     AND last_edited_date IS NOT NULL
5 )
6 and last_edited_date > DateAdd($(p_dateDimension), -$(p_dateValue), Convert(date, GetDate()))]
```

3. Next we expose the fme_feature_type attribute for grouping further downstream



2.1) Custom Transformer: ADOT_Filter

The custom ADOT_Filter transformer shown below has a single user parameter, allowing the user to SQL Wildcard select a specific route for testing or select all routes. It uses the wildcard to modularly build a sql select statement using attributes. This way, when values in fields like LocationError change that we need to reference, we can adjust the query in one spot and dynamically pass the query as an attribute to workspaces reading this database.

User Parameters for 'Main'

Route ID Match Pattern (Example '%I 040 %0%'):

Parameter Identifier: p_RouteID_Pattern

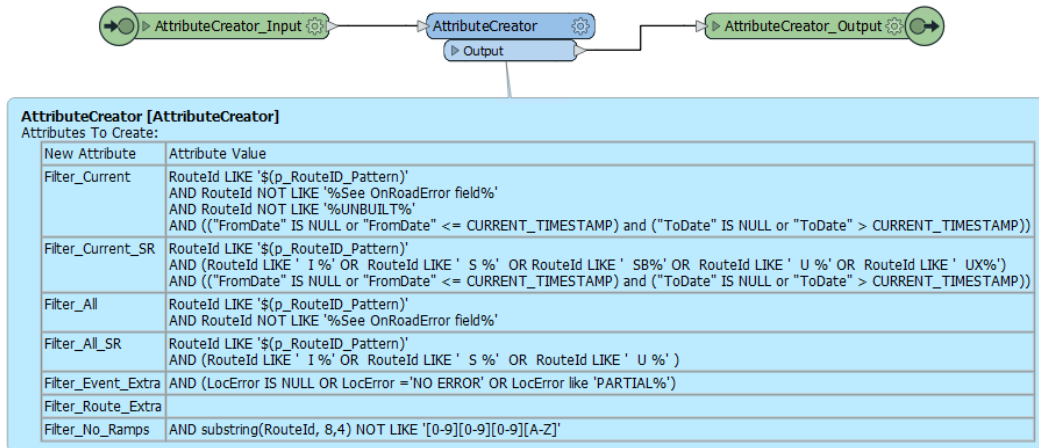
Prompt: Route ID Match Pattern (Example '%I 040 %0%')

☒ Published ☐ Required ☐ Disable Attribute Assignment

> Text Configuration

Default Value: %I 040%

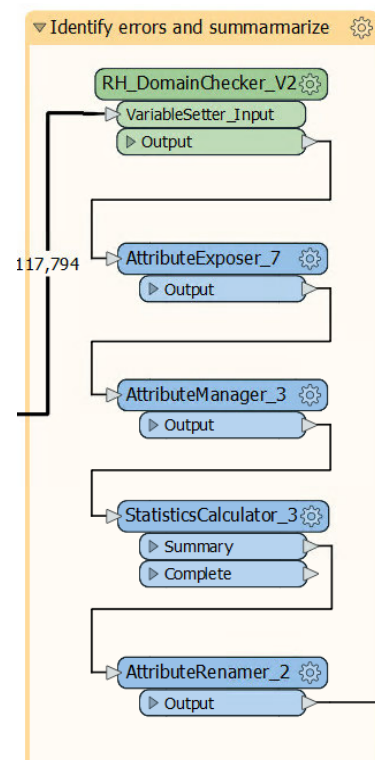
Preview... OK Cancel



3) Identify Errors and Format output

This process is fairly complex and ill break the custom workspace down after the main explanation as follow:

1. Pass all records into the RH domain Checker custom transformer
2. Expose output attributes to identify edit date and user for use in dashboard visualization
3. Use an attribute manager to rename fields and datetime format values in edit date
4. Generate statistics
5. Rename output of statistics calculator (would be very nice if this was an option in the transformer, to specify output field names!!!!)



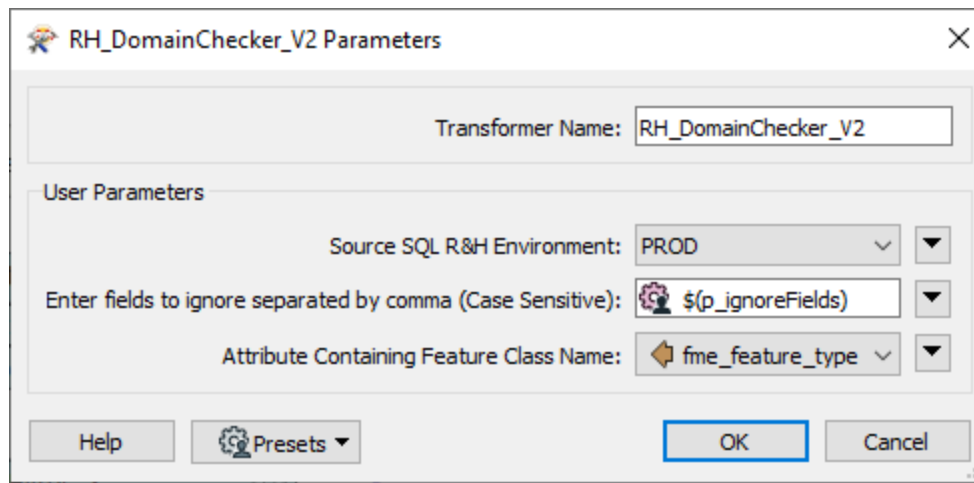
3.1) Custom Transformer: RH_DomainChecker_V2

Due to the immense size of the dataset this workspace process, many filtering steps have been applied in hopes of reducing processing time and throwing away irrelevant data as soon as possible. Lets take a step by step walkthrough of this process:

3.1.1) Input Parameters

The input parameters allow a user to:

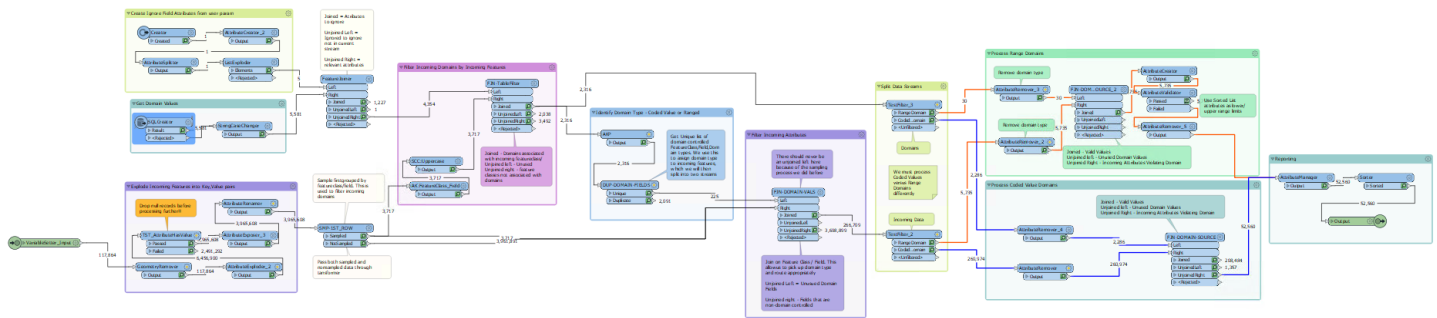
1. specify a target environment, development or production, using embedded database connection. I plan on migrating this to a shared database connection in a custom folder on the network drive in the future. Since this is a read only operation from the production database, anyone can have access rights to this connection on our team.
2. Provide a comma separated list of Field names to ignore. In this case, we are passing the parameter set in the main workspace dialog to this workspace so it's managed from the primary dialog box.
3. Specify the field name containing the feature class name. This is used for joining to domain tables.



3.1.2) Main Overview

The workspace consists on many parts which we will step through:

1. Reading domains from database
2. Applying user defined field filters
3. Exploding incoming features to key/value pairs
4. Filtering incoming domains by current field names in process. Remember that this is a time based selection, so not every domain will be needed in the process. Throwing them out here prevents having to join them against the full dataset later
5. Identifying coded value vs range value domains and join type to incoming feature steams
6. Separate streams
7. Process results
8. Report results



3.1.3) Reading Domains from the Database and Applying user Defined Filters

1. Generating Attributes for ignore fields:

- Here we use an attribute creator to create a single attribute using the user defined parameter of the ignore fields. The user supplies a comma separated list of values and an attribute is created from them. We then split the attribute into a list based on the comma, then we use a list exploder to generate a single attribute for each field to ignore

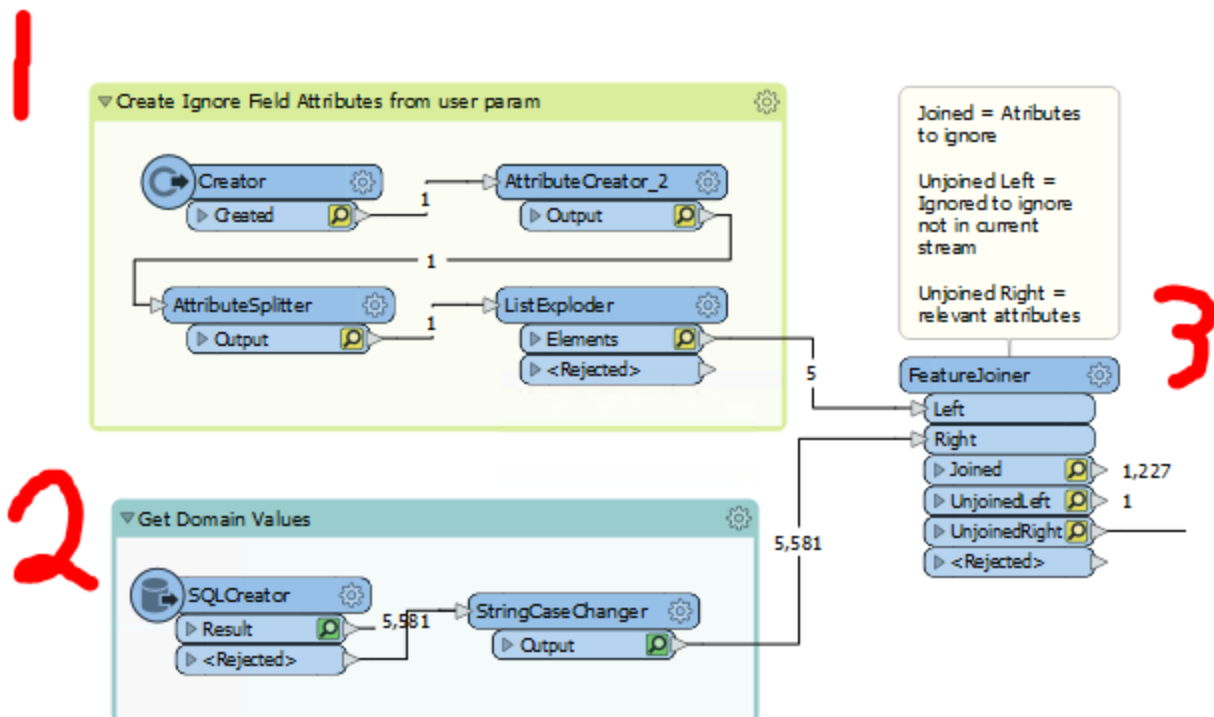
2. While the previous step is happening, we also:

- Initiate a sql query through a SQL creator to generate the Domain objects in the database.
- Force the table names to be uppercase for the subsequent feature joiner
- Query:

```
WITH ffd AS ( SELECT substring(i.Name, charindex('.', i.Name) + 1, 50) AS FeatureClass, xVal.value('Name[1]', 'nvarchar(max)') Field, xVal.value('DomainName[1]', 'nvarchar(max)') Domain FROM dbo.GDB_ITEMS i JOIN dbo.GDB_ITEMTYPES it ON i.Type = it.UUID CROSS APPLY i.Definition.nodes('/GPFieldInfoExs/GPFieldInfoEx') dx(xVal) WHERE i.NAME IS NOT NULL AND xVal.value('DomainName[1]', 'nvarchar(max)') IS NOT NULL ), dvc AS ( SELECT items.Name AS "Domain", itemtypes.Name AS "DomainValueType", codedValue.value('Code[1]', 'nvarchar(max)') AS "Code", codedValue.value('Name[1]', 'nvarchar(max)') AS "Value", null AS "ValueRange" FROM dbo.GDB_ITEMS AS items INNER JOIN dbo.GDB_ITEMTYPES AS itemtypes ON items.Type = itemtypes.UUID CROSS APPLY items.Definition.nodes('/GPCodedValueDomain2/CodedValues/CodedValue') AS CodedValues(codedValue) WHERE itemtypes.Name = 'Coded Value Domain' ), dvr AS ( SELECT items.Name AS "Domain", itemtypes.Name AS "DomainValueType", null AS "Code", null AS "Value", items.Definition.query('/GPRangeDomain2/MinValue').value('MaxValue[1]', 'nvarchar(10)') + ' ' + items.Definition.query('/GPRangeDomain2/MaxValue').value('MaxValue[1]', 'nvarchar(10)') AS "ValueRange" FROM dbo.GDB_ITEMS AS items INNER JOIN dbo.GDB_ITEMTYPES AS itemtypes ON items.Type = itemtypes.UUID WHERE itemtypes.Name = 'Range Domain' ) SELECT FeatureClass, field, dvc.domain, dvc.DomainValueType, dvc.Code, dvc.Value, dvc.ValueRange FROM dvc JOIN ffd ON dvc.DOMAIN = ffd.domain AND dvc.domain NOT IN ('dReferentMethod') UNION ALL SELECT FeatureClass, field, dvr.domain, dvr.DomainValueType, dvr.Code, dvr.Value, dvr.ValueRange FROM dvr JOIN ffd ON dvr.DOMAIN = ffd.domain
```

3. Filtering incoming domains based on ignore fields

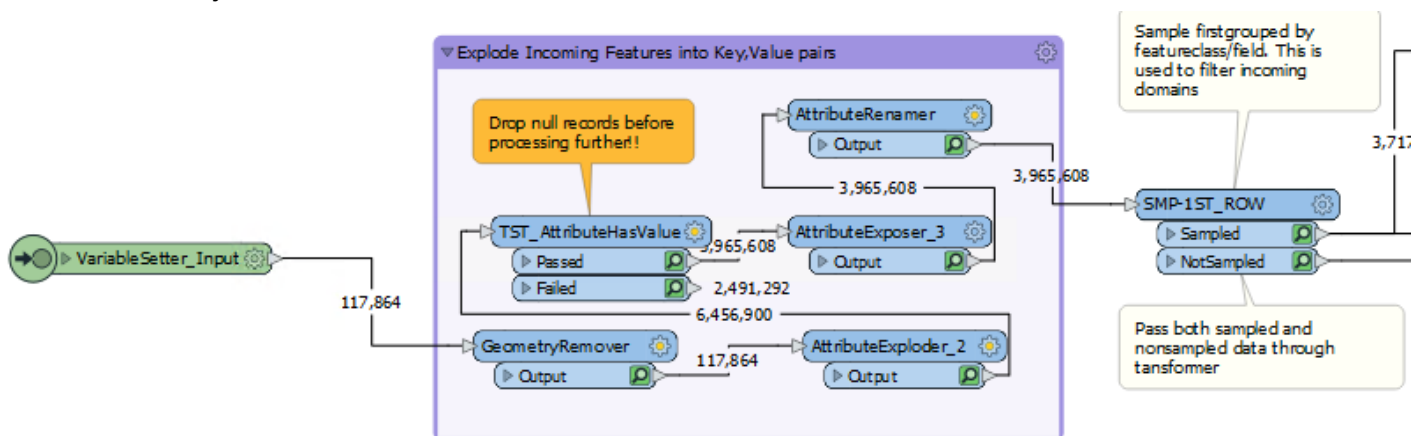
- Here we simply use a feature joiner to join the Incoming Domain Fields to the user specified Ignore Fields.



3.1.5) Exploding incoming features to key/value pairs

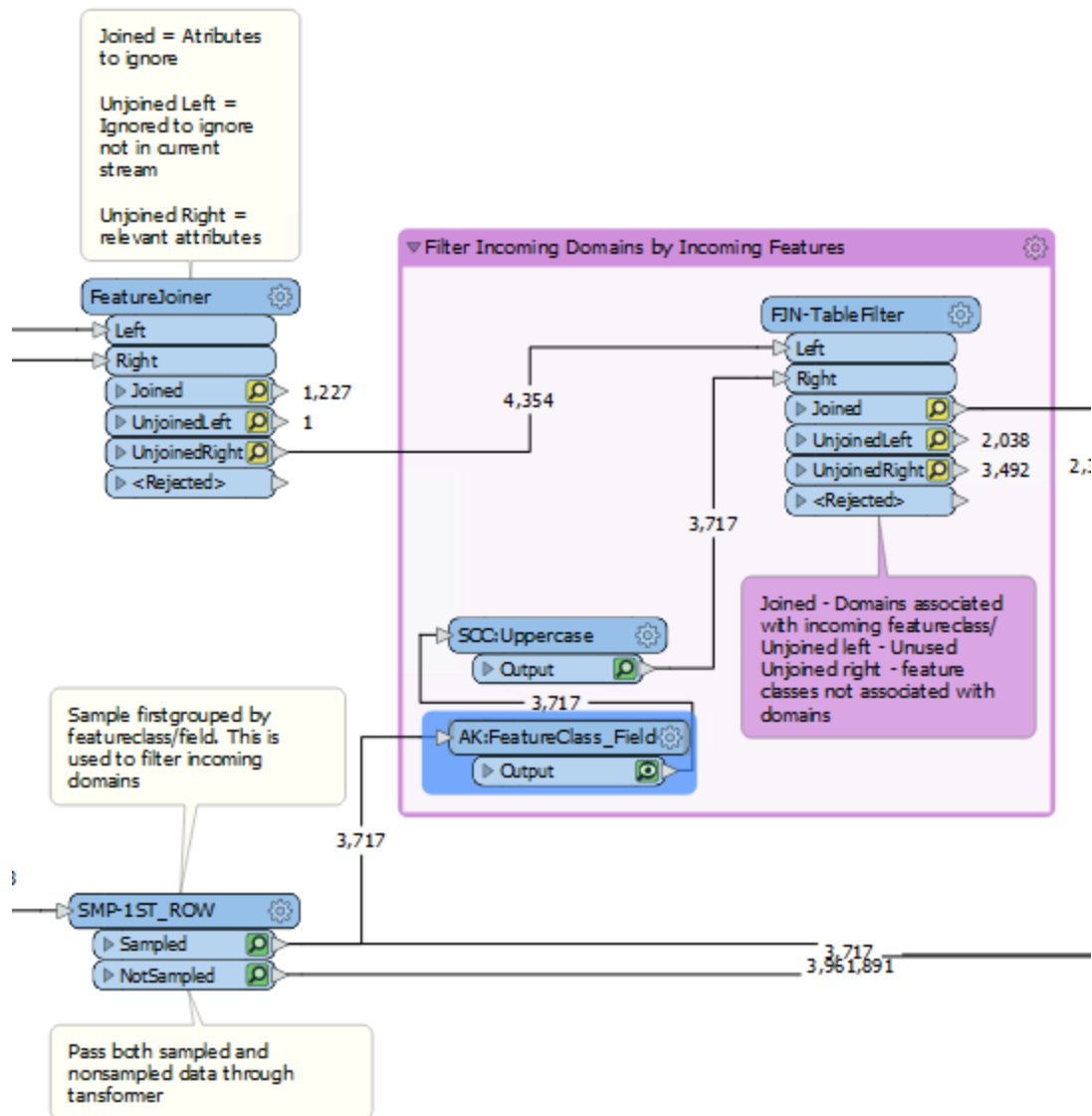
While the Domain aggregation and preliminary filtering is happening above, we also process the incoming record stream:

1. Remove geometry if there is any, this is an attribute based report.
2. Explode the attributes into key values pairs. This gives us an attribute for Field/Value for each record. This will be used to join the domains for validation later.
3. Next we test if the value is null, drop all records who don't have a value.
4. Now we expose fme_feature_type, which in this case is the name of the origin feature class. We now have a FeatureClass/Field/Value attribute
5. Rename attributes for consistency and joining later
6. Now we sample the 1st record grouped by FeatureClass/Field. This will allow us to further widdle down the incoming domains for joining.
 - a. This step is imperative for performance. Instead of joining 4 million records to 5000 records, we can join 4million records to 250 records



3.1.6) Filtering incoming domains by incoming features

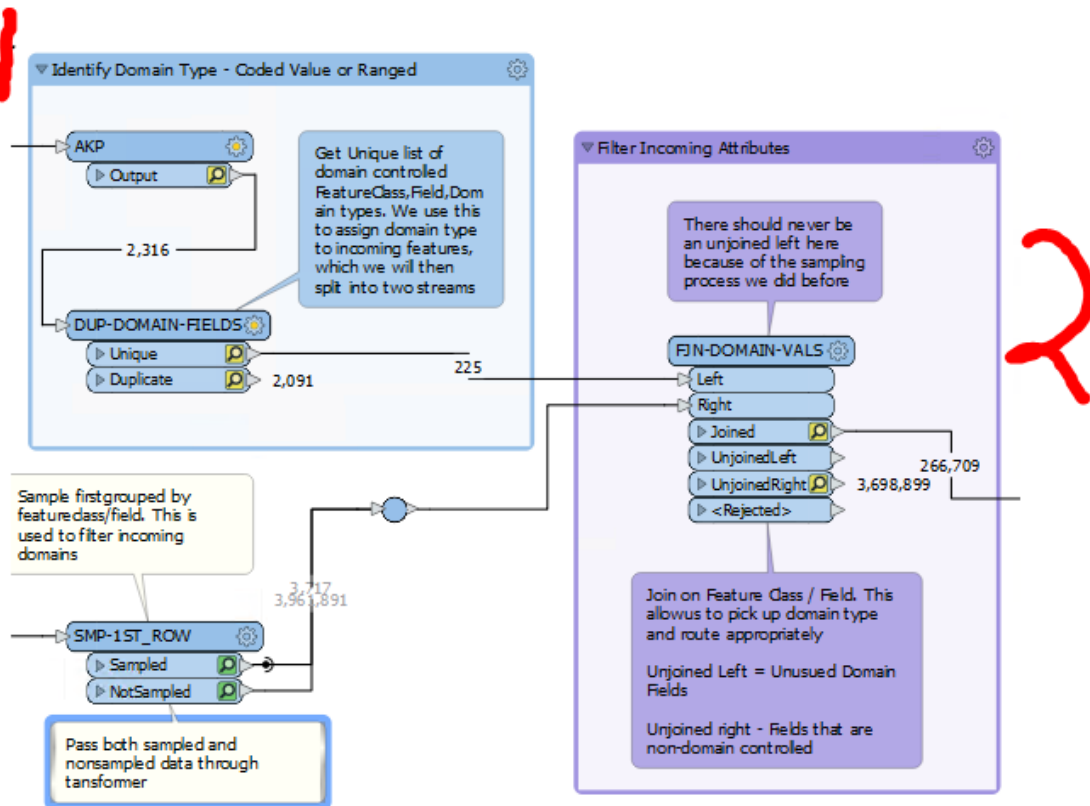
As previously mentioned, in order to improve performance we drop any irrelevant attributes as we can. The next step in this process is to use the output of the feature joiner and the sampler from the previous steps to further filter the domain values. We take our incoming domain attributes which have been filtered once (removing any fields the user specified to ignore), and we join that to the unique value list of FeatureClass/Field. This step removes any domain values that apply to fields that are not in the current data stream. Simultaneously, we pass all records (both sampled and unsampled) through to the next step because the sampled records were just used to filter the incoming domains.



3.1.7) Identifying coded value vs range value domains

Next, I:

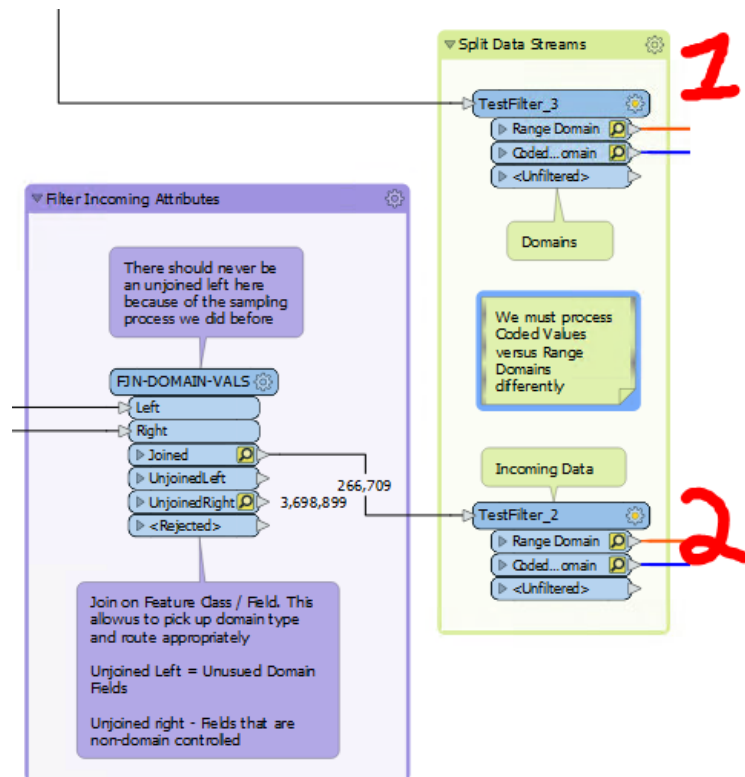
1. take the output of the feature joiner and keep three attributes: FeatureClass Name, Domain Value Type (coded/range), and the FieldName. I then use a duplicate filter to generate a unique list of these three fields.
2. Join the incoming data stream to the unique list of FeatureClass/Field/Domain type. This allows me to assign the CodedValue or RangeValue type to each incoming field/value pair for proper processing.



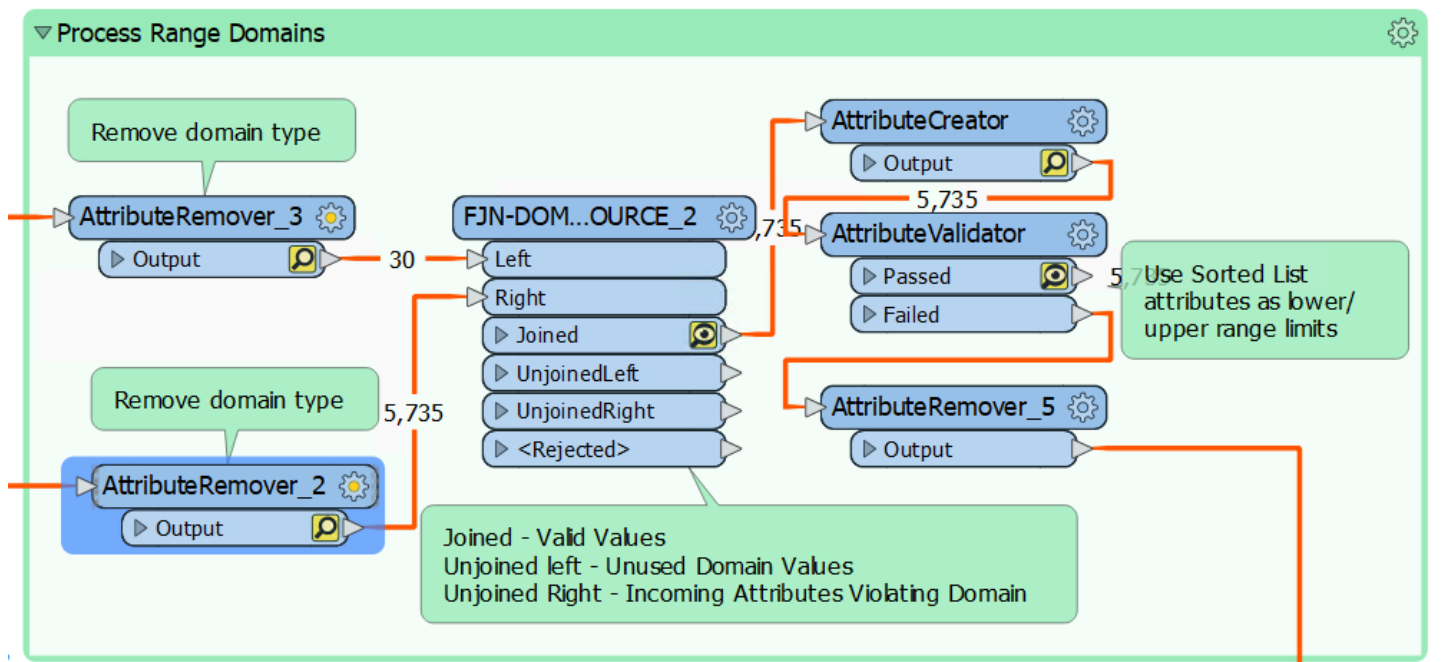
3.

3.1.8) Separate streams

1. The output of the feature joiner in the “*Filter Incoming Domains by Incoming Features*” bookmark is carried through to this test filter annotated as “domains” for proper routing to QC Processes. Coded values and range values have to be processed differently.
2. The incoming data stream is also split into two streams, crange domain and coded domain, for processing. This is possible because of the previous join we did to carry through attributes from the domain stream onto the attribute stream.



3.1.9) Process Range Domains



First we remove the domain type from both incoming streams. At this point our domain stream should have a FeatureClass Name, a Field Name, and a Value Range. Out Data Stream should have a Feature Class name, Field Name, and Field Value

Domains

FeatureClass	ValueRange	field
DBO.LRSE_AADT	1900, 2050	SubmittalYear

Data Stream

FeatureClass	field	_attr_value
DBO.LRSE_AADT	SubmittalYear	2020

We take both of these and join them together giving us a Feature Class, Field, Valid Value Range, and Value attributes:

FeatureClass	ValueRange	field	_attr_value
DBO.LRSE_AADT	1900, 2050	SubmittalYear	2020

Finally, we can use an attribute validator to ensure our value is within the value range:

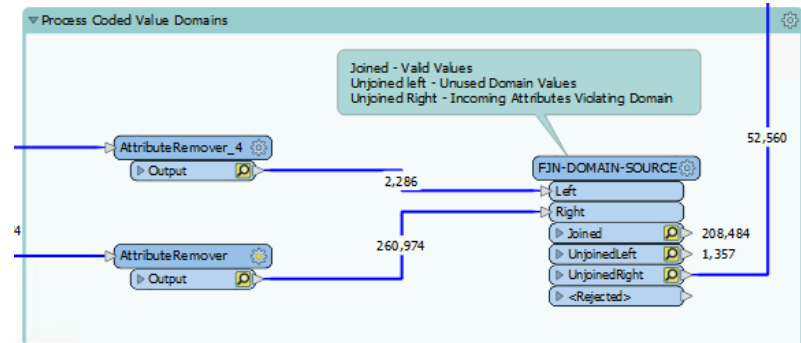
Transformer Name:

Validation Rules

Attributes to Validate	Validation Rule	Rule Configuration
☐ _attr_value	In Range	ValueRange

3.1.10) Process Coded Domains

For coded domains, the process is much simpler. Take the incoming domain stream, and the incoming data stream, and join them together based on FeatureClass, Field, Value. Whatever falls out of this join is a value in the data table that doesn't exist in the domain table.



Domain Stream

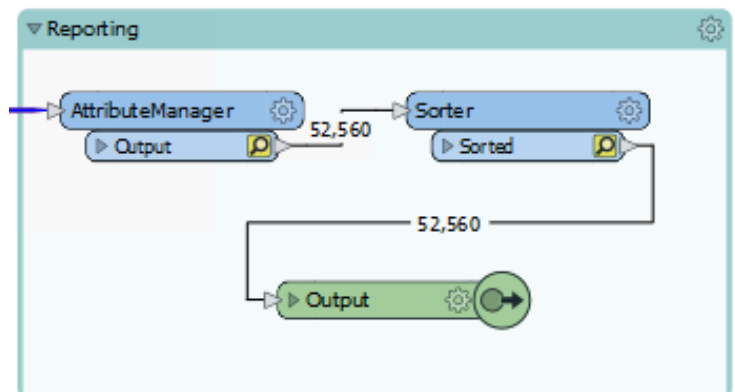
FeatureClass	Code	field
DBO.LRSE_ACCE...	1	AccessControl

Data Stream

FeatureClass	field	_attr_value
DBO.LRSE_ACCE...	AccessControl	3

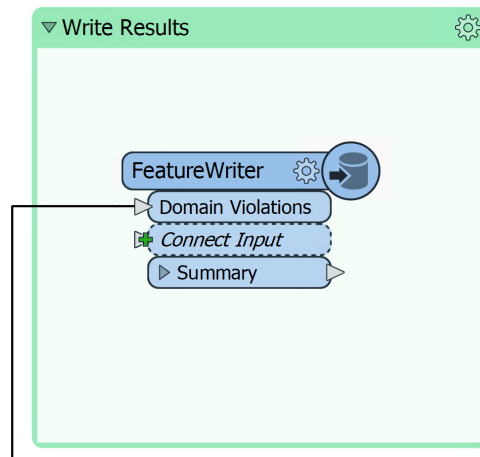
3.1.13) Report results

Finally, we adjust the final output with an attribute manager and then sort the results.



4) Write Results

In the version I'm submitting for review, I have changed the output to a .xlsx file to also include the .xlsx file for your review. However in the main production process, the results are actually written to a google sheet for display in a google dashboard which I will include screenshots of. The writer is configured to truncate and insert into the spreadsheet.



This process I am displaying is actually 1/3 of a much larger process designed to provide an authoritative dashboard for management at ADOT to provide a holistic overview of their Roads and Highways database. The dashboard gives insight into which contractors are inputting errors into the database, how frequently these errors are happening, and where they're occurring. This dashboard provides a great near-real-time feedback mechanism for ADOT to review personnel training and associated internal editing processes. Additionally, I have left a few bookmarks out of the workspace i'm submitting to avoid complicating it further, but these additional bookmarks are what populate the reports editor names/ companies, as well as generating the QC summary tab.

The google sheet consists of 4 tabs:

- This tab is for errors specific to the Domain Workspace. This the the direct data source used to power the various widgets of the domain report on the dashboard

RH QC Dashboard Data Sources

File

Edit

View

Insert

Format

Data

Tools

Add-ons

Help

Last edit was 5 days ago

100%

\$ % . _ 123

Default (Ari...

10

B I U A

</

QC Summary Tab

The QC summary logs each resulting runs input and outputs for future analysis. It is important to know what datasets we excluded in prior runs because it can help us determine if a drastic change in error record counts is associated with a parameter change, a staffing change, or just a one time large scale anomaly.

RH QC Dashboard Data Sources

docs.google.com/spreadsheets/d/1gFtN6rexkuMTV1trTx09jGZVnRQ5xNpG5CggLlGPc/edit?pli=1#gid=669971827

RH QC Dashboard Data Sources

File Edit View Insert Format Data Tools Add-ons Help Last edit was 5 days ago

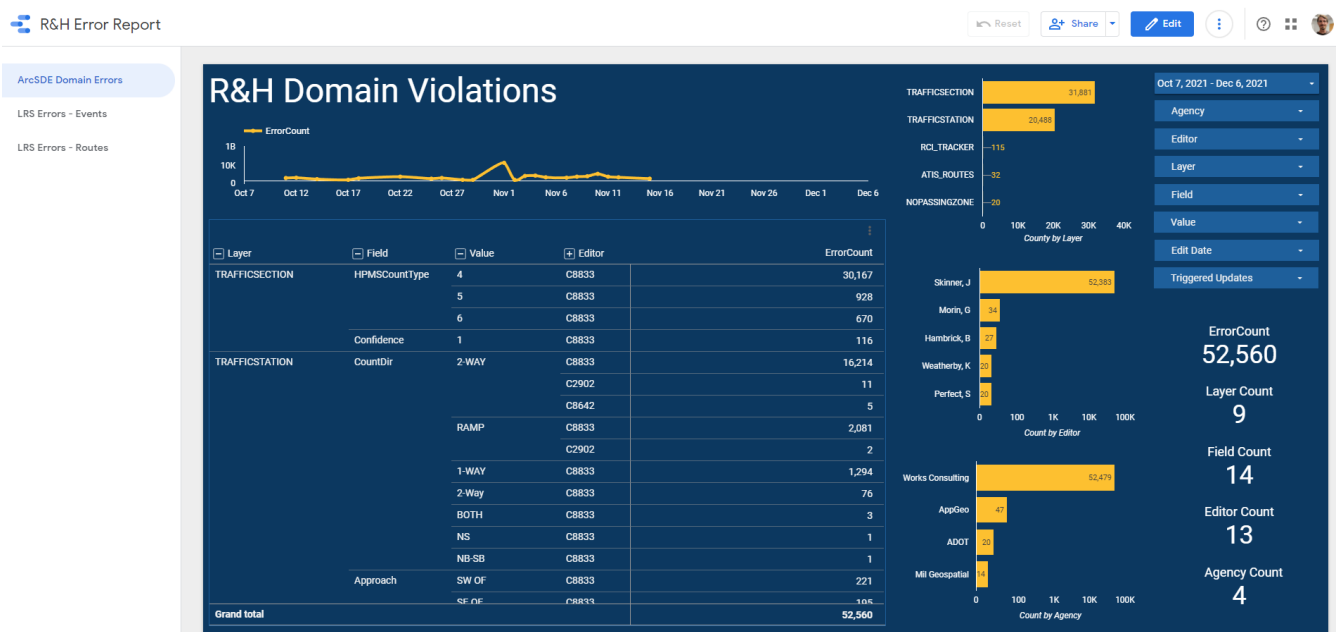
100% 123 Default (Arial) 12

SHARED PARAMS								DOMAIN SPECIFIC				EVENT SPECIFIC						ROUTE SPECIFIC	
QCDate	Date Dimension	Date Value	Triggered Update Offset	Route Filter	Missing Table Emails	Missing Users Emails	IgnoreFields_d	IgnoreLayers_d	TotalCount_d	ErrorCount_d	IgnoreLayers_e	MinEvent Length	Precision	Gap Checked	Overlap Checked	TotalCount_e	ErrorCount_e	TotalCount_r	ErrorCount_r
20211123	dd	60	2 %	John.ehleri	John.ehleri	John.ehleri	DataSource,Sou	None	126101	52598	AADT_CPMMASTI	5	5	CARRIAG	AADT, AAT	117474	4082	84	139
20211201	dd	60	2 %	John.ehleri	John.ehleri	John.ehleri	DataSource,Sou	None	120336	52561	AADT_CPMMASTI	5	5	CARRIAG	AADT, AAT	118001	2210	76	119

Data Studio Report

While I can't grant you access to the dashboard, I can show you a screenshot of it as seen below. Because this report is intended for managers and not data scientists, I have included many easy to use metrics and charts to allow simple drill downs for data visualization and exploration. The Dashboard user can select a time range for errors (by default we build this dashboard weekly, looking back 60 days), Agency (ADOT Vs individual contractors), Editor, Layer (Feature Class), Field, Field Value, Specific Edit Dats, and whether or not the edit was a triggered update*.

- The triggered update logic has been removed from the simplified workspace, but it is an additional bookmark that attempts to make an educated best-guess as to whether this record was updated manually or whether the update was triggered to an event layer based on an edit to its associated route in the LRS Network.



More Info

If you're interested in learning more about this Project 2, feel free to watch either of the presentations I gave on it:

Arizona Geographic Information Council, 2021 - [Navigating Roads to Data Visualization, AGIC 2021 - John Ehlen](#)

Esri Roads and Highways User Group, 2021 - [Visualizing Esri Roads & Highways Errors, RHUG 2021 - John Ehlen](#)

Additionally, I manage and host the FME In Transportation Special Interest Group on LinkedIn.

<https://www.linkedin.com/groups/12505249/>

We have a semi monthly presentation series bringing together FME Experts in the Transportation section from around the country to show off various tip, tricks, and best practices in using FME for LRS related processes.